# CHMOD

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]



2007-03-19

## Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 7739 bytes

| | |
|---|---|
| **Attack Category** | • Path spoofing or confusion problem |
| **Vulnerability Category** | • TOCTOU - Time of Check, Time of Use<br>• Indeterminate File/Path<br>• Privilege escalation problem |
| **Software Context** | • File Management |
| **Location** | |
| **Description** | The chmod() and fchmod() functions set the access permission portion of the mode of the file whose name is given by path or referenced by the open file descriptor files to the bit pattern contained in mode. This function is used to change the read/write permissions of a file.<br><br>Note: The functions of the chmod class have significantly differing functionality and warrant individual description.<br><br>lchmod(), while having the same function signature as chmod, differs from chmod in that it does not follow symbolic links.<br><br>fchmod(), while performing the same function as chmod, operates on a file descriptor, and not a filename.<br><br>chmod() is vulnerable to TOCTOU attacks. The existence of a call to this function should be flagged regardless of whether a check function precedes it. |
| **APIs** | **FunctionName** / **Comments**<br>_chmod<br>_tchmod<br>_wchmod<br>chmod<br>lchmod<br>fchmod |

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

| | |
|---|---|
| **Method of Attack** | The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results.<br><br>The chmod() call is a use-category call, which when preceded by a check-category call can be indicative of a TOCTOU vulnerability. In particular, the fact that chmod() uses a name allows the attacker to replace that named resource between the time of check and use.<br><br>A TOCTOU attack in regards to chmod() can occur when a check for the existence of a file occurs (or some other check function) and then chmod() occurs. Between those two actions, an attacker could, for example, link the target file (the one to be chmod'ed) to a different known file. The subsequent chmod of the target file would result in modifying the characteristics of the attacked file. |
| **Exception Criteria** | |
| **Solutions** | |

| Solution Applicability | Solution Description | Solution Efficacy |
|---|---|---|
| Generaly applies to any chmod(). | Don't use chmod. Use fchmod if the check function has an fd version. | Effective. |
| Generally applies to any chmod(). | The most basic advice for TOCTOU vulnerabilities is to not perform a checkl before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but | Does not resolve the underlying vulnerability but limits the false sense of security given by the check. |

| | | | |
|---|---|---|---|
| | | it does help to limit the false sense of security given by the check. | |
| | Generally applies to any chmod(). | Limit the interleaving of operations on files from multiple processes. | Does not eliminate the underlying vulnerability but can help make it more difficult to exploit. |
| | Generally applies to any chmod(). | Limit the spread of time (cycles) between the check and use of a resource. | Does not eliminate the underlying vulnerability but can help make it more difficult to exploit. |
| | Generally applies to any chmod(). | Recheck the resource after the use call to verify that the action was taken appropriately. | Checking the status after the operation does not change the fact that the operation may have been exploited but it does allow halting of the application in an error state to help limit further damage. |

| | |
|---|---|
| **Signature Details** | int chmod(const char *path, mode_t mode);<br>int lchmod(const char *path, mode_t mode);<br>int _wchmod(const wchar_t *path, mode_t mode);<br>int fchmod(int fildes, mode_t mode); |
| **Examples of Incorrect Code** | `#include <sys/types.h>`<br>`#include <sys/stat.h>`<br><br>`int check_status;`<br>`int use_status;`<br>`struct stat statbuf;`<br>`mode_t Mode;`<br>`...`<br><br>`check_status=stat("thefile",statbuf);`<br><br>`...`<br>`<long enough intervening code>` |

| | |
|---|---|
| | `use_status=chmod("thefile",Mode);` |
| **Examples of Corrected Code** | One solution is to eliminate the check test and instead check status afterwards.<br>`#include <sys/types.h>`<br>`#include <sys/stat.h>`<br>`int status;`<br>...<br>`status=chmod(path, S_IRUSR\|S_IRGRP\|S_IROTH);`<br>`/* check the status .... */` |
| **Source References** | • Viega, John & McGraw, Gary. Building Secure Software: How to Avoid Security Problems the Right Way. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X<br>• chmod man page<br>• Kruegel, Christopher. Secure Software Programming and Vulnerability Analysis[2]. |
| **Recommended Resource** | |
| **Discriminant Set** | **Operating Systems**: • UNIX • Windows<br>**Languages**: • C • C++ |

# Cigital, Inc. Copyright



The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com